Olsker Cupcakes

Mike Patrick Nørlev Andersen
Email: cph-ma813@cphbusiness.dk
Github: zPaddyz
Hold A

Lars
Email: cph-lg195@cphbusiness.com
Github: LaCabraGrande
Hold B

Lukas
Email: cph-lr278@cphbusiness.dk
Github: LukasRonberg
Hold A

Ingrid
Email: cph-is142@cphbusiness.dk
Github: Ingridksv
Hold B

Projekt start: 2 April 2024
Projekt slut: 11 April 2024

Indholdsfortegnelse

Indledning

Målgruppen for denne rapport er en datamatiker studerende på 2. semester.

Dette projekt omhandler en hjemmeside der skulle laves til Olsker Cupcakes. Formålet med hjemmesiden er at kunden/brugeren skal kunne vælge en topping, en bund og et antal af den valgte kombination, hvorefter man skal kunne lægge den i sin kurv og betale for de forskellige kombinationer af toppings og bunde man har valgt.

Video gennemgang af hjemmesidens funktionalitet (11-04-2024):

https://cphbusiness.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=6063263d-92fe-4e63-88d8-b14f0075fc91

Teknologivalg

Vi har valgt de anviste teknologier, som vi har arbejdet med igennem datamatikeruddannelsen. Udover det, har vi også valgt at bruge JavaScript for at få et mere responsivt design med pop up bokse.

Github
Github Kanban
Java v8
JavaScript
Maven v3.10.1
PostgreSQL v42.7.2
Docker v4.28.0
PgAdmin4 v8.3
Intellij Ultimate v3.3.5
Thymeleaf v3.1.2.RELEASE
Javalin v6.1.3

Krav

Projektets user stories

US-1: Som kunde kan jeg bestille og betale cupcakes med en valgfri bund og top, så jeg senere kan køre forbi butikken i Olsker og hente min ordre.

US-2 Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en ordre.

US-3: Som administrator kan jeg indsætte beløb på en kundes konto direkte i Postgres, så en kunde kan betale for sine ordrer.

US-4: Som kunde kan jeg se mine valgte ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris.

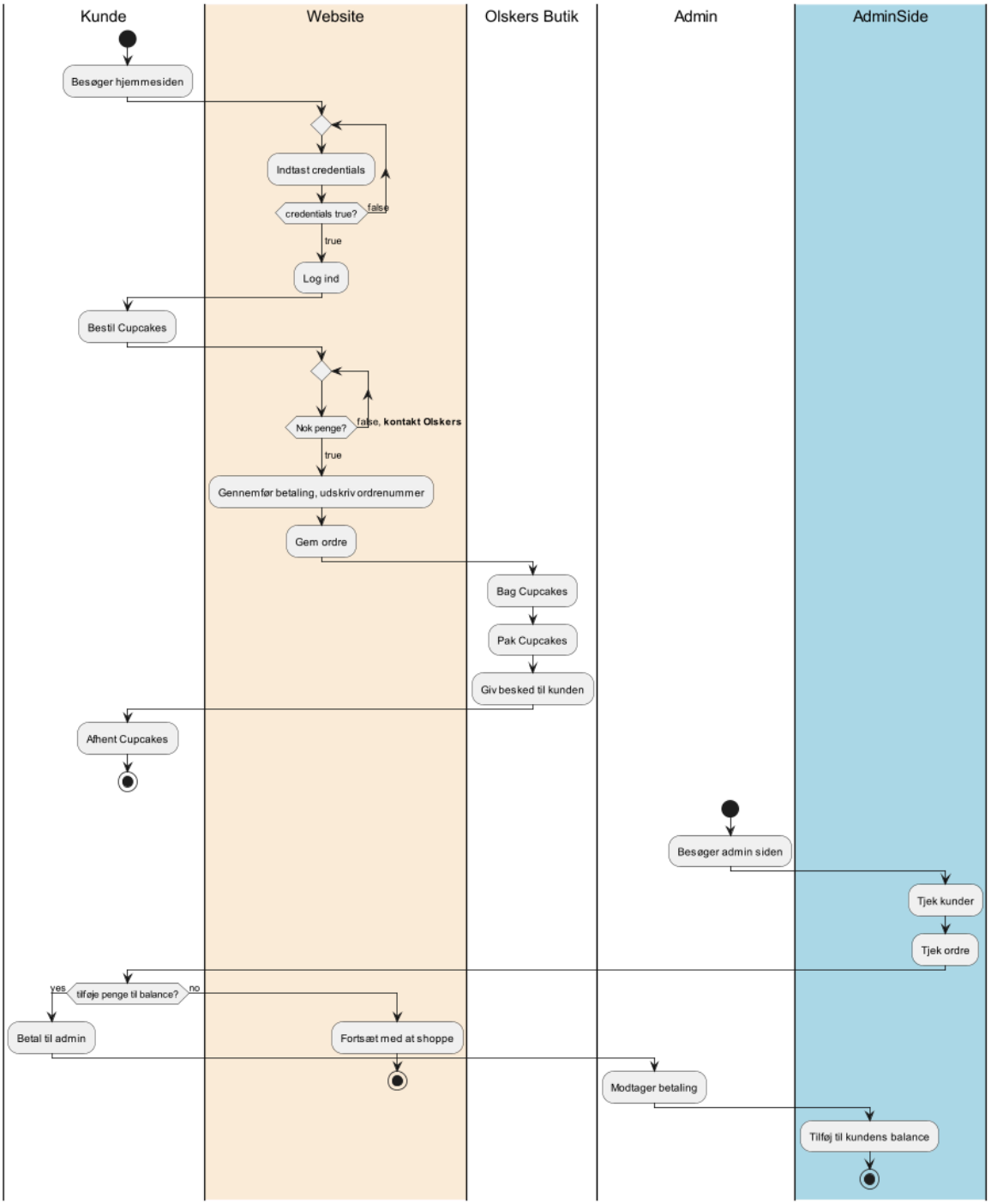
US-5: Som kunde eller administrator kan jeg logge på systemet med e-mail og kodeord. Når jeg er logget på, skal jeg kunne se min email på hver side (evt. i topmenuen, som vist på mock up'en).

US-6: Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.

US-7: Som administrator kan jeg se alle kunder i systemet og deres ordrer, så jeg kan følge op på ordrer og holde styr på mine kunder.

US-8: Som kunde kan jeg fjerne en ordrelinje fra min indkøbskurv, så jeg kan justere min ordre.

Aktivitetsdiagram

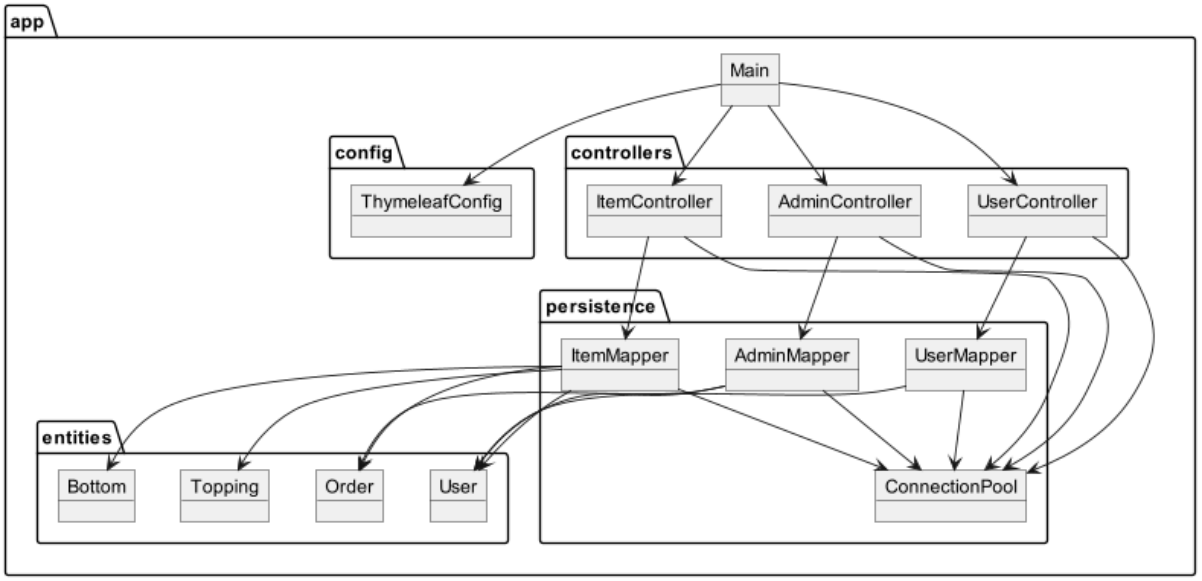


Aktivitetsdiagrammet er opdelt i tre svømmebaner, der viser kundens, hjemmesidens og Olskers roller igennem hele processen fra kunden besøger hjemmesiden til afhentning af cupcakesene. Når kunden besøger hjemmesiden, skal denne først logge ind, før cupcakesene bliver indsat i kurven. Herefter bestiller kunden de valgte cupcakes og hjemmesiden vil derefter tjekke om kunden har nok penge, hvis dette ikke er tilfældet, så

skal kunden kontakte Olskers for at få sat penge ind.
Har kunden penge nok, så skriver hjemmesiden et ordrenummer samt en tak for bestillingen. Ordren gemmes i databasen og Olskers kan begynde at bage, pakke samt at give besked til kunden, at deres ordre er klar til afhentning. Kunden afslutter processen med at afhente ordren.

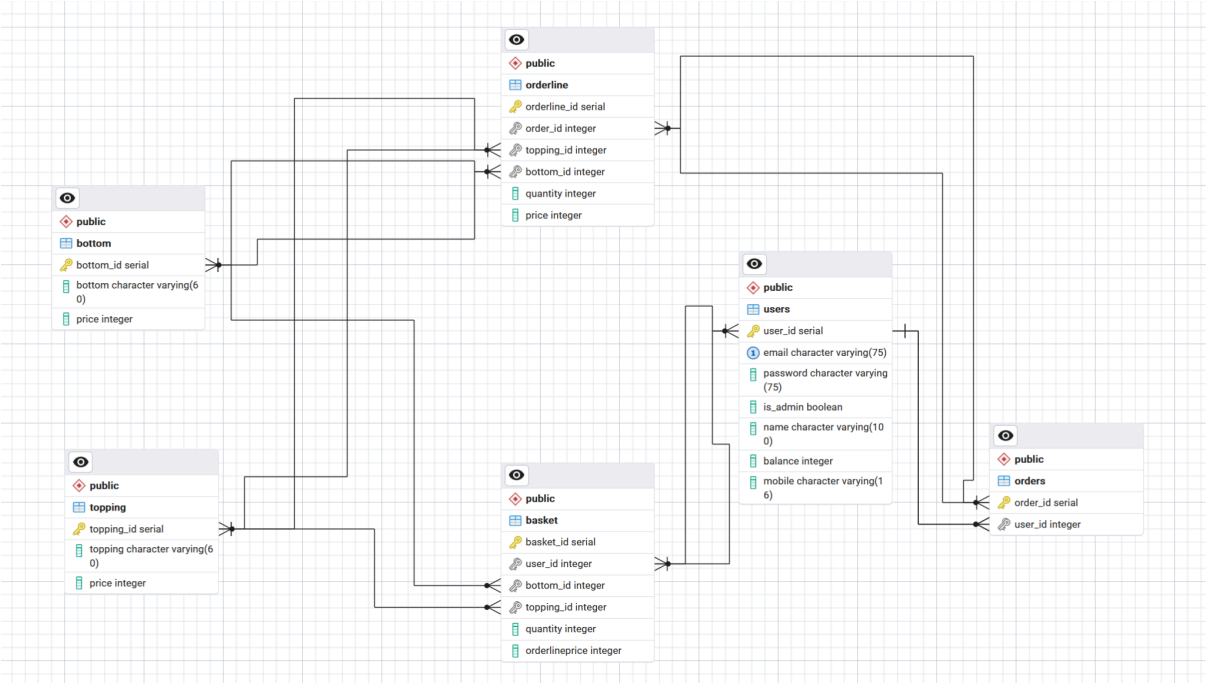
Admin kan tjekke kunderne og deres ordrer og kan tilføje penge til kundens konto.

Domænemodel



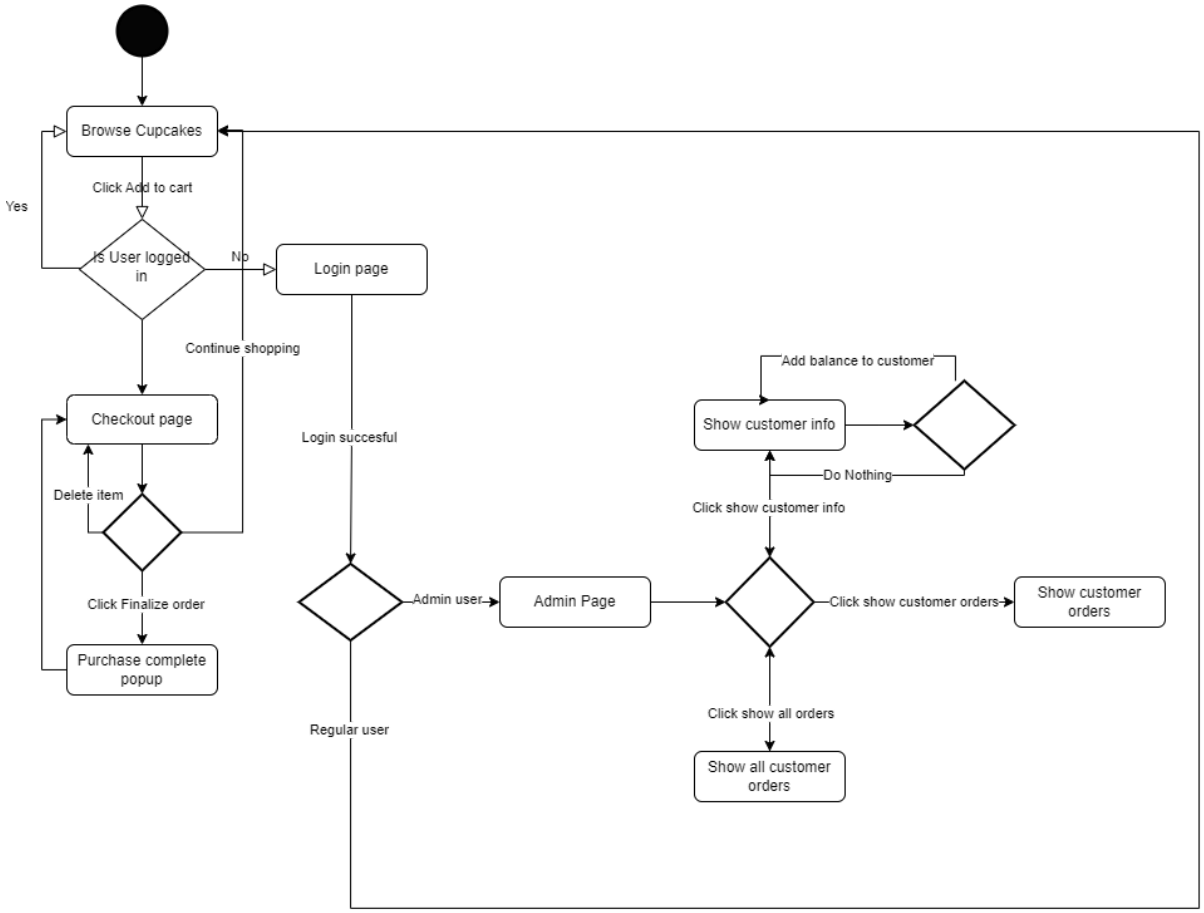
Denne domænemodel beskriver relationerne der er imellem de forskellige klasser vi har i vores og program, så man kan se hvordan vi har valgt at strukturere projektet.
Vores fremgangsmåde har været at lave flere forskellige mapper til de forskellige klasser for at få et bedre overskud over projektet. De mapper vi har lavet er controllers, persistence og entities mapper.
Controller klasserne er bindeleddet mellem mapperne og HTML'en. Den tager input fra mapper klasserne og håndterer GET og POST requests.
Mapper klasserne sørger for at hente den data vi skal bruge fra databasen.
Entities klasserne bruger vi til at lave objekter ud fra den data der er i databasen.

ER diagram



Ud fra opgavebeskrivelsen har vi valgt at løse opgaven med 5 tabeller i vores database. orderline og orders skriver vi til når en betaling bliver gennemført og det er også her vi henter oplysninger på vores adminSite. Bottom og Topping hiver vi oplysninger fra på vores forside(index.html). I den forbindelse lægger vi lister ind som sessionAttributter som vi så gennemløber med Thymeleaf kommandoer. Basket bruger vi til at gemme kundens kurv hvis han ikke har betalt for den. I login-metoden henter vi kundens kurv og lægger den i sessionattributten, og når vi logger ud, skriver vi kurven ud i tabellen som enkelte ordrelinier. Den eneste unikke atribut vi har er emailen i tabellen, når vi bruger emailen som brugernavn.

Navigationsdiagram



Dette navigationsdiagram viser processen igennem programmet som I også vil se i den korte, vedhæftede video. Processen starter ved forsiden af hjemmesiden hvor man har mulighed for at navigere i de forskellige typer toppings og bunde. Her er der mulighed for at trykke på en login knap samt tilføje ting til sin kurv. For en bruger som ikke endnu er logget ind, vil disse to knapper have den samme funktion, nemlig at dirigere til login-siden. På login-siden kan man oprette en bruger eller logge ind på en eksisterende bruger. Den almindelige bruger vil derefter blive sendt tilbage til forsiden hvor der nu er mulighed for at tilføje ting til sin kurv. Når det ønskede antal af cupcakes er tilføjet, vil brugere kunne trykke på kurv ikonet i højre hjørne, som vil transportere brugeren til checkout siden. På denne side har brugeren mulighed for at slette tidligere tilføjet cupcakes og fuldføre sit køb.
Såfremt brugere har sig en admin bruger vil denne i stedet blive dirigeret til admin panelet hvor der er mulighed for at søge på nuværende brugere, se deres ordre eller se alle brugerens ordre. Ved søgning på en bestemt bruger vil admin have muligheden, at tilføje penge til dennes balance.

Særlige forhold

- Vi bruger en del sessionAttributter til at opdatere de forskellige data på siderne. Attributten orders er en liste over kundens ordrelinier. TotalAmount er den totale pris på alle ordrelinierne. orderCount beskriver antallet af ordrelinierne. login, notenoughmoney og ordercreated bruges når vi skal vise de forskellige popup-javascript.
- Vi har ikke valgt at lave validering på brugerinputtet ved oprettelse af en bruger.
- Når en bruger bliver logget ind opretter vi en sessionAttribut ved navn currentUser.
- Vi har en boolean attribut som hedder is_admin. Giver vi værdien true, er det en admin. False er det en normal kunde.

```
private static void createOrder(Context ctx, ConnectionPool
connectionPool) throws DatabaseException {
    User currentUser = ctx.sessionAttribute("currentUser");
    if (currentUser == null) {
        ctx.render("login.html");
        return;
    }

    if (ctx.sessionAttribute("orders") != null) {
        orderLine = ctx.sessionAttribute("orders");
    }

    String email = currentUser.getEmail();
    String name = currentUser.getName();
    String mobile = currentUser.getMobile();
    int balance = currentUser.getBalance();

    int toppingId = Integer.parseInt(ctx.formParam("topping")); //
Assumes you have toppingId in form
    int bottomId = Integer.parseInt(ctx.formParam("bund"));    //
Assumes you have bottomId in form
    int quantity = Integer.parseInt(ctx.formParam("antal"));
```

```
    Topping topping = ItemMapper.getToppingById(toppingId,
connectionPool);
    Bottom bottom = ItemMapper.getBottomById(bottomId,
connectionPool);
    if (topping == null || bottom == null) {
        throw new DatabaseException("cant find id for bottom or
topping");
    }

    int orderlinePrice = calculateOrderLinePrice(topping, bottom,
quantity);

    Order order = new Order(currentUser.getUserId(), email, name,
mobile, balance, topping.getTopping(), bottom.getBottom(),
quantity, orderlinePrice);
    orderLine.add(order);

    ctx.sessionAttribute("orders", orderLine);


    int totalAmount = 0;
    int orderCount = 0;
    for (Order orderline : orderLine) {
        if (order.getUserId() == currentUser.getUserId()) {
            totalAmount += orderline.getOrderlinePrice();
            orderCount++;
        }
    }

    ctx.sessionAttribute("totalAmount", totalAmount); // Sender det
samlede beløb som en attribut til HTML-skabelonen
    ctx.sessionAttribute("orderCount", orderCount);

    showTopping(ctx, ConnectionPool.getInstance());
    showBottom(ctx, ConnectionPool.getInstance());
    ctx.render("index.html");
}
```

I dette eksempel er det funktionen der tilføjer en cupcake til brugerens kurv, der bliver vist. Dette sker ved at der først bliver hentet den nuværende bruger af programmet og såfremt der ikke er nogen, vil der blive dirigeret videre til login-siden. Når den nuværende bruger eksisterer, vil programmet kigge efter orders attributen, som indeholder de tidligere ordrer afgivet. Programmet vil derefter hente den nuværende brugers email, navn, telefonnummer og balance. Derefter hentes der toppingId, bottomId og quantity fra de angivet html forms. Herefter henter vi den topping og bottom der er blevet valgt, da vi skal bruge dem som parametre i calculateOrderLinePrice funktionen. Variablen orderlinePrice bliver så sat ved hjælp af denne hjælper funktion.
I næste trin bliver der oprettet et order objekt, som tager alle de tidligere deklarerede variable som parametre. Så tilføjes den nye ordre til den tidligere hentet ordreliste “orders” som så igen bliver sat til at være en sessionattribute og på den måde overskriver den gamle. Næste trin er udregning af totalprisen for alle cupcakes i kurven og det samlede antal af unikke typer cupcakes, der kan findes i kurven. Dette sker ved hjælp af et for loop der checker at ordrens bruger id og den nuværende brugeres id stemmer overens derefter lægges den tidligere udregnede orderlinePrice til total prisen og ordercount variablen tælles op således at antallet kan vises ved siden af kurven. Disse to variabler bliver så gemt som sessionattributer så de kan bruges til fremvisning i html siden. Som sidste trin bliver showTopping og showBottom kørt da vi henter vores toppings og bottoms fra databasen er dette nødvendigt hver gang vi skal rendere index.html siden.

Status på implementation

Produktets endegyldige status er for teamet, yderst tilfredsstillende, der er opnået 90% af kundens ønsker i forhold til programmets funktionalitet, nedenunder er der nævnt ting som stod i backlog og/eller som var ønskelige at have med i programmet.

- Ens styling på hovedside og admin side
- Check således at almene brugere ikke kan tilgå admin side
- Gemning af kurv uden at være logget ind
- Korrekt exception håndtering - specielt ved input
- Fjernelse af ordre fra administratorside
- Rettelse/fjernelse/oprettelse af toppings og bunde på administratorside
- Unit Tests er ikke blevet lavet
- Manglende optimering
- Mulighed for at se ens saldo
- Mulighed for at se tidligere bestillinger og status på disse.
- Validering af e-mail, telefonnummer, og navne.

Proces

Teamets arbejdsform var baseret på en tildeling af opgaver igennem github's kanban, heri blev der oprettet opgaver fra start, som blev anset for at være nødvendige for et MVP. Teamet uddelte/påtog sig hver i sær opgaver som individet følte at man selv kunne varetage. Senere i forløbet blev brugen af kanban mindre og mindre, da større dele af projektet blev udviklet i plenum. Teamet er tilfreds over samarbejdet og føler at projektet er lykkedes. Til næste projekt har teamet gjort sig nogle iagttagelser som vil være gode at huske på, og som vil kunne bidrage til et endnu bedre projekt samt samarbejde.
Herunder indgår, strengere gennemgang af pull-requests, længere diskussion omkring opbygning af database, forventet slut dato på grundlæggende funktionaliteter, aftaler og enighed omkring programmets endelige funktioner og bedre prioritering af arbejdskraft.